Разбор задач

Задача 1. Кодовый замок

Нам нужно составить строку наименьшей длины из цифр «1», «2», «3», содержащую в качестве подстрок все возможные строки длины 3 из этих же цифр. Таких подстрок существует 27, поэтому ответ должен иметь длину не менее 29 символов: первые 3 символа дают одну подстроку длины 3, добавление каждого следующего символа добавляет ещё одну подстроку.

Пример строки длины 29, удовлетворяющей условию задачи: 11121131222123132133322323311.

Опишем возможный способ получения такой строки. Рассмотрим граф из 9 вершин, соответствующих подстрокам длины 2: «11», «12», «13», «21», «22», «23», «31», «32», «33». Соединим вершины рёбрами, из каждой вершины будут выходить 3 ребра, одно ребро соответствует одной добавляемой цифре. Ребро ведёт в вершину, которой сопоставлены последние 2 символа полученной последовательности. Например, из вершины «12» ребро с цифрой «1» ведёт в вершину «21», ребро «2» ведёт в вершину «22», а ребро «3» ведёт в вершину «23». Проход по одному ребру соответствует подстроке длины 3 (две цифры — это последовательность, записанная в вершине, третья цифра записана на ребре). Чтобы все возможные строки длины 3 встретились, нужно построить путь, содержащий все рёбра. Поскольку граф связный, в каждую вершину входит ровно 3 ребра и выходит ровно 3 ребра, то можно построить замкнутый путь, содержащий все рёбра ровно по одному разу, он и будет соответствовать кратчайшей искомой последовательности.

Задача 2. Билеты в кинотеатр

Для маленьких значений n ответ можно получить моделированием процесса вручную. К первому кассиру покупатели будут подходить каждые 30 секунд, то есть через 0, 30, 60, 90, 120, 150, ... секунд после начала продажи билетов. Ко второму кассиру покупатели будут подходить через 0, 50, 100, 150, ... секунд. К третьему — через 0, 75, 150, ... секунд. Если объединить все эти списки вместе и упорядочить их, то мы получим время ожидания для разных значений n.

n	Ответ
0	0
1	0
2	0
3	30
4	50
5	60
6	75
7	90
8	100
9	120
10	150
11	150
12	150

Через 150 секунд освободятся все три кассира, так как 150 является наименьшим общим кратным чисел 30, 50, 75. За это время кассиры вместе обслужат 10 покупателей. И далее последовательность повторяется, с добавлением 150 секунд при каждом увеличении значения n на 10. Поэтому ответ для $n = 33$ равен $3 \cdot 150 + 30 = 480$, а ответ для $n = 89$ равен $8 \cdot 150 + 120 = 1320$.

Для вычисления ответа для больших значений n можно использовать электронные таблицы или написать программу.

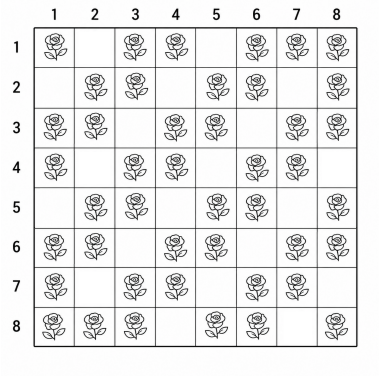
Ответ:

60
75
150
480
1320
1440

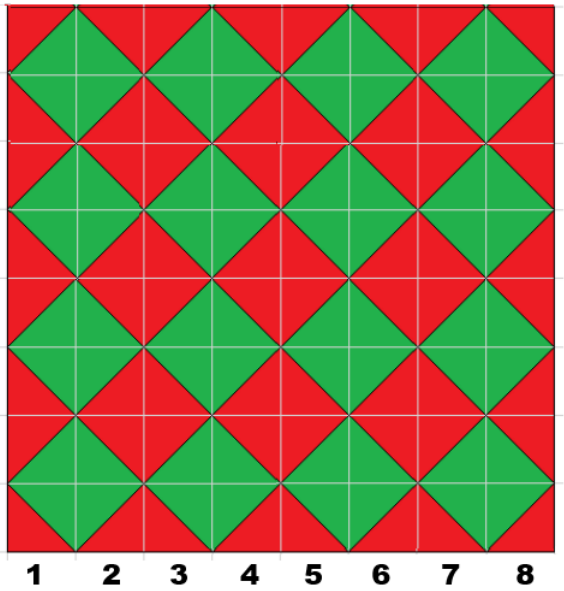
1800
4150
5750
8550
12990
15050
68500
106425
125850
440773800
458875410
1010774250
1274073480
1499999970

Задача 3. Розарий

Наибольшее количество цветов, которое можно разместить в розарии, равно 44. На рисунке приведён один из вариантов решения.



Задача 4. Аргайл



Первая подзадача:

Если посмотреть на получившийся узор, то можно заметить, что новый ряд зелёных квадратов появляется при каждом чётном n, а новый ряд красных — на каждом нечётном, начиная с 3. Таким

образом, можно перебрать все промежуточные значения, увеличивая счётчик рядов на 1 при каждом выполнении условия чётности или нечётности для определённого цвета. Такое решение наберёт 50 баллов.

```
n = int(input())
red = 0
for i in range(2, n + 1):
    if i % 2 == 1:
        red += 1
green = 0
for i in range(1, n + 1):
    if i % 2 == 0:
        green += 1
print(red ** 2)
print(green ** 2)
```

Полное решение.

Воспользуемся операциями целочисленного деления. Один зелёный квадрат располагается на квадрате 2×2, поэтому ответ для этого цвета будет равен (n // 2) ** 2.

Если убрать полосы шириной 1 слева и снизу ткани, то ответ для красного цвета сведётся к предыдущему: теперь один красный квадрат располагается на квадрате 2 на 2, поэтому ответ для этого цвета будет равен ((n - 1) // 2) ** 2.

```
n = int(input())
red = ((n - 1) // 2) ** 2
green = (n // 2) ** 2
print(red)
print(green)
```

Задача 5. Оптом — дешевле?

Прежде всего отметим, что всегда выгодно купить коробку вместо k отдельных упаковок сока. Поэтому при необходимости приобрести определённое количество упаковок сока, нужно посчитать количество полных упаковок, взяв частное от деления на k, и остаток от деления на k будет равен количеству отдельных упаковок, которое необходимо приобрести.

Рассмотрим сначала случай, когда хотя бы одна из величин n или m делится нацело на k. Для определённости будем считать, что $n = \text{boxesN} \cdot k$, а $m = \text{boxesM} \cdot k + \text{packsM}$ (при необходимости следует поменять местами n и m в рассуждениях).

Это значит, что на апельсиновый сок друзья должны потратить $boxesN \cdot p$ рублей, а на оставшиеся деньги они приобретут максимально возможное количество упаковок яблочного сока. Если это количество окажется не меньше m, то условия задачи будут выполнены.

Пример решения.

```
n = int(input())  # Требуется первого сока
m = int(input())  # Требуется второго сока
r = int(input())  # Цена упаковки сока
k = int(input())  # Количество упаковок сока в коробке
p = int(input())  # Цена коробки сока
s = int(input())  # Сумма денег

if n % k != 0:
    n, m = m, n
boxesN = n // k   # коробок первого сока

rem = s - boxesN * p  # останется денег после приобретения первого сока
boxes = rem // p  # количество коробок, купленное на оставшиеся деньги
```

```
packs = (rem % p) // r  # количество упаковок, купленное на оставшиеся деньги

packsRem = boxes * k + packs  # суммарное количество упаковок, купленное на оставшиеся деньги

if packsRem >= m:
    ans = n + packsRem
else:
    ans = -1
print(ans)
```

Для полного решения заметим, что нет разницы, на какой вид сока потратить излишек денег. Давайте считать, что первого сока мы купим столько, сколько необходимо, а излишки потратим на покупку максимального количества второго сока. Но возможны два способа приобрести первый сок: приобрести ровно n упаковок целыми коробками и отдельными упаковками или вместо отдельных упаковок приобрести одну дополнительную коробку сока. На оставшиеся деньги приобретается максимальное количество второго сока, как в предыдущем решении.

Пример решения на языке Python.

```
n = int(input())  # Требуется первого сока
m = int(input())  # Требуется второго сока
r = int(input())  # Цена сока
k = int(input())  # Кол-во сока в коробке
p = int(input())  # Цена упаковки
s = int(input())  # Сумма денег
ans = -1

boxesN = n // k   # коробок первого сока
packsN = n % k    # упаковок первого сока

rem = s - boxesN * p - packsN * r  # останется денег после приобретения n упаковок первого сока
if rem >=0:
    boxes = rem // p
    packs = (rem % p) // r
    if packs + boxes * k >= m:
        ans = max(ans, packsN + boxesN * k + packs + boxes * k)
boxesN += 1                        # добавим одну коробку первого сока
packsN = 0                         # а упаковки приобретать не будем
rem = s - boxesN * p  - packsN * r

if rem >= 0:
    boxes = rem // p
    packs = (rem % p) // r
    if packs + boxes * k >= m:
        ans = max(ans, packsN + boxesN * k + packs + boxes * k)

print(ans)
```

Задача 6. Арт-объект

При $n = 1$ последовательность из одной колонны уже неубывающая, поэтому нужно вывести 0.

При $n = 2$, если $a_0 > a_1$, то ответ будет равен $a_0 - a_1$, иначе ответ 0. Разобрав эти два случая, можно набрать 30 баллов.

Если высоты колонн небольшие, то можно перебрать значения d. Прибавив к значениям a_i нужные значения, проверим, что мы получили неубывающую последовательность.

Такие решения будут набирать не менее 30 баллов. Пример такого решения.

```
n = int(input())
a = [int(input()) for i in range(n)]
d = 0
while True:
    i = 1
    while i < n and a[i - 1] <= a[i]:
        i += 1
    if i == n:
        print(d)
        break
    for i in range(n):
        a[i] += i + 1
    d += 1
```

Для того чтобы набрать полный балл, можно заметить, что если разница между двумя соседними значениями $a_i - a_{i+1}$ положительна, то ответ должен быть не меньше этой разницы. Поэтому минимальное значение ответа будет равно наибольшему из значений $a_i - a_{i+1}$. Этот максимум можно вычислить сразу при считывании данных, необязательно сохранять все значения a_i в массиве, достаточно запоминать только значение высоты предыдущей колонны.

Пример такого решения.

```
n = int(input())
ans = 0
prev = 0
for i in range(n):
    curr = int(input())
    ans = max(ans, prev - curr)
    prev = curr
print(ans)
```